Energy minimization via graph-cuts

Nikos Komodakis
Ecole des Ponts ParisTech, LIGM

Traitement de l’information et vision artificielle
Binary energy minimization

- We will first consider binary MRFs:
  - Graph is arbitrary (not necessarily a tree)
  - Only 2 labels per node

- We will use graph-cut techniques to optimize their energy
Binary MRFs

- We will consider each pairwise potential as a non-negative distance function between labels
  - Pairwise potential = \[
  \begin{array}{cc}
  0 & w_{pq} \\
  w_{pq} & 0 \\
  \end{array}
  \]

- Energy for resulting binary MRFs:
  \[
  \sum_p V_p(x_p) + \sum_{pq \in N} w_{pq} \cdot \delta(x_p \neq x_q)
  \]
Binary MRFs

- We will try to reduce this to a graph-cut problem

- But first let's take a look at what we mean by a graph-cut
Graph-cuts basics
Graph-cuts basics

Goal: divide the graph into two parts separating orange and green nodes

A graph with two terminals \( S \) and \( T \)

- Cut cost is a sum of severed edge weights
- Minimum cost \( s-t \) cut can be found in polynomial time
Graph-cuts basics
Graph-cuts basics

- All methods for solving min-cut actually rely on solving another problem, called max-flow.

- Max-flow problem can be shown to be equivalent to min-cut.
The Maximum Flow Problem

- Directed Graph \( G = (N, A) \).
  - Source \( s \)
  - Sink \( t \)
  - Capacities \( u_{ij} \) on arc \((i,j)\)
  - Maximize the flow out of \( s \), subject to:

\[
\text{Flow out of } i = \text{Flow into } i, \text{ for } i \neq s \text{ or } t.
\]

A Network with Arc Capacities and Flows
Residual network

- Plays a central role in the development of maximum flow algorithms.

- Defined with respect to a flow $x$.

- Denotes how much flow can be sent on arcs with respect to a flow $x$. 
We let $r_{ij}$ denote the residual capacity of arc (i,j)

The Residual Network $G(x)$
Max-flow algorithms

Two classes of algorithms exist for solving this problem:

- Augmenting paths
- Push-relabel
Augmenting Paths

- An **augmenting path** is a path from s to t in the residual network.

- The **residual capacity** of the augmenting path P is $\delta(P) = \min\{r_{ij} : (i,j) \in P\}$.

- To **augment along P** is to send $d(P)$ units of flow along each arc of the path. We modify $x$ and the residual capacities appropriately.

- $r_{ij} := r_{ij} - \delta(P)$ and $r_{ji} := r_{ji} + \delta(P)$ for $(i,j) \in P$. 
Augmenting Paths algorithm

- Find an augmenting path from S to T
- Increase flow along this path until some edge saturates

A graph with two terminals
Augmenting Paths algorithm

- Find an augmenting path from $S$ to $T$
- Increase flow along this path until some edge saturates
- Find next path...
- Increase flow...

A graph with two terminals
Augmenting Paths algorithm

- Find an augmenting path from S to T
- Increase flow along this path until some edge saturates

A graph with two terminals

Iterate until … all paths from S to T have at least one saturated edge

MAX FLOW ↔ MIN CUT
Augmenting Paths algorithm

- Max-flow saturates min-cut

- Suppose we have solved the max-flow problem. How do we extract a min-cut?

- To obtain a min-cut, we simply need to consider all nodes reachable from “source” in the residual network $G(x)$
The Ford Fulkerson Maximum Flow Algorithm

Begin

- $x := 0$;
- create the residual network $G(x)$;
- while there is some directed path from $s$ to $t$ in $G(x)$ do
  - begin
    - let $P$ be a path from $s$ to $t$ in $G(x)$;
    - $\Delta := \delta(P)$;
    - send $\Delta$ units of flow along $P$;
    - update the $r$'s;
  - end
- end {the flow $x$ is now maximum}. 
Ford-Fulkerson Max Flow

This is the original network, and the original residual network.
Ford-Fulkerson Max Flow

Find any s-t path in G(x)
Ford-Fulkerson Max Flow

Determine the capacity $\Delta$ of the path.

Send $\Delta$ units of flow in the path.
Update residual capacities.
Ford-Fulkerson Max Flow

Find any s-t path
Ford-Fulkerson Max Flow

Determine the capacity $\Delta$ of the path.

Send $\Delta$ units of flow in the path.

Update residual capacities.
Ford-Fulkerson Max Flow

Find any s-t path
Ford-Fulkerson Max Flow

Determine the capacity $\Delta$ of the path.

Send $\Delta$ units of flow in the path. Update residual capacities.
Find any s-t path
Determine the capacity $\Delta$ of the path.

Send $\Delta$ units of flow in the path. Update residual capacities.
Ford-Fulkerson Max Flow

Find any s-t path
Determine the capacity $\Delta$ of the path.

Send $\Delta$ units of flow in the path. Update residual capacities.
Ford-Fulkerson Max Flow

There is no $s$-$t$ path in the residual network. This flow is optimal.
Ford-Fulkerson Max Flow

These are the nodes that are reachable from node s.
Ford-Fulkerson Max Flow

Here is the optimal flow
Optimizing binary MRFs

Let us now return to our original goal, i.e., the minimization of the following binary MRF energy:

$$\sum_p V_p(x_p) + \sum_{pq \in N} w_{pq} \cdot \delta(x_p \neq x_q)$$

How can we reduce this to a min-cut problem?

To understand how this can be done, we will concentrate on some simple cases first.
Special case 1:

- Let us further assume that:
  - The MRF graph is a 2D grid with N nodes $x_1, \ldots, x_N$
  - The unary potentials are defined as follows:

$$V_1(x_1) = \begin{cases} +\infty, & \text{if } x_p = 0 \\ 0, & \text{if } x_p = 1 \end{cases} \quad V_N(x_N) = \begin{cases} 0, & \text{if } x_p = 0 \\ +\infty, & \text{if } x_p = 1 \end{cases}$$

All other unary potentials are assumed to be zero.

- What does the resulting MRF problem represent?
- How can we reduce this case to a graph-cut problem?
Special case 2:

- What about if we had (not just two) but multiple nodes with infinite potentials?

- What is the corresponding weighted graph in this case?
Case 3:

- Given how we reduced the binary MRF problem to a min-cut in the previous two simple cases, can we now do the same for our original binary energy?

\[ \sum_{p} V_p(x_p) + \sum_{pq \in N} w_{pq} \cdot \delta(x_p \neq x_q) \]

- What is the min-cut graph in this case?
Binary segmentation
Binary segmentation

- Simplest case is to just set the edge weights that define the pairwise potentials (*boundary terms only*)
Binary segmentation

- More generally, for the segmentation problem we can make use of:
  - the pairwise potentials (i.e., the edge weights) to model **boundary terms**
  - the unary potentials to model **regional terms**
Binary segmentation

- Boundary extraction (in 2D image space)
Binary segmentation

- Boundary extraction (in 3D volume space)
Binary segmentation

- Boundary extraction (in 3D volume space)
Binary segmentation

- A simple toy case for the regional terms is if we know that foreground/background are assumed to have some “expected” intensities or colors.

\[
V_p(x_p) = \begin{cases} 
|I_p - I_{background}| \\
|I_p - I_{foreground}|
\end{cases}
\]
Binary segmentation

- Of course, more complex intensity models can be used for the foreground/background
  - Histograms

\[ V_p(x_p) = -\ln \Pr(I_p | x_p) \]
Binary segmentation

- Of course, more complex intensity models can be used for the foreground/background
  - Histograms
Binary segmentation

propagation in 3D space
Binary segmentation

propagation in space and time
Binary segmentation

- Other types of intensity models for foreground/background
  - Mixture of gaussians

- Adaptive estimation of intensity models
  - Iterative algorithms for image segmentation (iterated cuts)
  - E.g., GrabCut is one such example
Binary segmentation

- One can also use more complex models for the pairwise interactions as well

- Texture Gradient $TG(x,y,r,\theta)$
  - $\chi^2$ difference of texton histograms
  - Textons are vector-quantized filter outputs